ARMY RESEARCH LABORATORY

yScalable, a Code for Calculating Probability of Incapacitation for Multi-Round Munitions

by Robert J. Yager

ARL-TR-6297 **January 2013**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-TR-6297 **January 2013**

yScalable, a Code for Calculating Probability of Incapacitation for Multi-Round Munitions

Robert J. Yager
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2013	Final	March 2012–September 2012

4. TITLE AND SUBTITLE
yScalable, a Code for Calculating Probability of Incapacitation for Multi-Round Munitions

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Yager

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-WML-A
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-6297

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
yScalable is a stand-alone C++ code that can be used to calculate Probability of Incapacitation (PI) values for a barrage of multi-round munitions deployed against a fixed set of personnel targets. Functions and methodologies that were developed to evaluate high-explosive-munition lethality and collateral damage have been adapted and extended to provide analytic capabilities that didn't previously exist.

15. SUBJECT TERMS
CASRED, C++ probability of incapacitation, multiple detonation warheads

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert J. Yager
Unclassified	Unclassified	Unclassified	UU	30	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6689

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

The author would like to thank Mr. Richard Pearson of the U.S. Army Research Laboratory's Weapons and Materials Research Directorate for his technical and editorial recommendations.

INTENTIONALLY LEFT BLANK.

1. Introduction

yScalable is a stand-alone C++ code that can be used to calculate Probability of Incapacitation (PI) values for a barrage of multi-round munitions deployed against a fixed set of personnel targets. Functions and methodologies that were developed to evaluate high-explosive-munition lethality and collateral damage[1] have been adapted and extended to provide analytic capabilities that didn't previously exist.

2. Definitions

To avoid confusion, several terms have been chosen to describe the different types of projectiles and projectile groups that are associated with a multi-detonation warhead. This set of definitions is not universally accepted and, thus, should be used with caution outside the scope of this report.

Fragment: A fragment is a piece of metal that has been produced as the result of the detonation of an explosive projectile.

Round: A round is a device that contains a single explosive charge and produces fragments when detonated.

Munition: A munition is a gun-launched projectile that may contain multiple rounds.

Barrage: A barrage is a set of munitions that are intended to work together to incapacitate a single target set, where a single target set consists of multiple personnel targets.

Iteration: The yScalable program uses a Monte Carlo method to calculate probabilities. As is the case with all Monte Carlo methods, random sampling is used to vary the outcome of a given scenario. Each repetition of the scenario is referred to as an iteration.

[1]Oberle, W.; Butler, P. *Development of a Performance Model to Evaluate High-Explosive Munition Lethality and Collateral Damage*; ARL-TR-4932; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2009.

3. Simplifying Assumptions

- For a given iteration, the timing between barrages is assumed to be short enough that the postures and locations of personnel targets do not change.
- The locations of personnel targets are known exactly.
- Although the delivery error that is associated with munitions does affect round placement, there is no delivery error that is specific to the rounds within a munition.

4. Methodology

For each iteration of a Monte Carlo run, two values are calculated.

1. The total number of target personnel that were incapacitated, and
2. The distance from the origin to the detonation point of the round that landed the farthest from the origin.

The above two values are used to calculate four values for the entire Monte-Carlo simulation.

1. The average number of personnel that were incapacitated per iteration,
2. The average probability of incapacitation,
3. The probability of incapacitating 100% of all targets, and
4. The average maximum distance from the origin to a detonation.

5. Geometry

As shown in figure 1, two Cartesian coordinate systems are used by the yScalable program. A primary coordinate system (shown in blue) is used to specify target locations and munition reference locations. Secondary coordinate systems (shown in red) are used to specify round-detonation locations. Both coordinate systems are assumed to lie in the ground plane, thus all locations are specified in two dimensions.

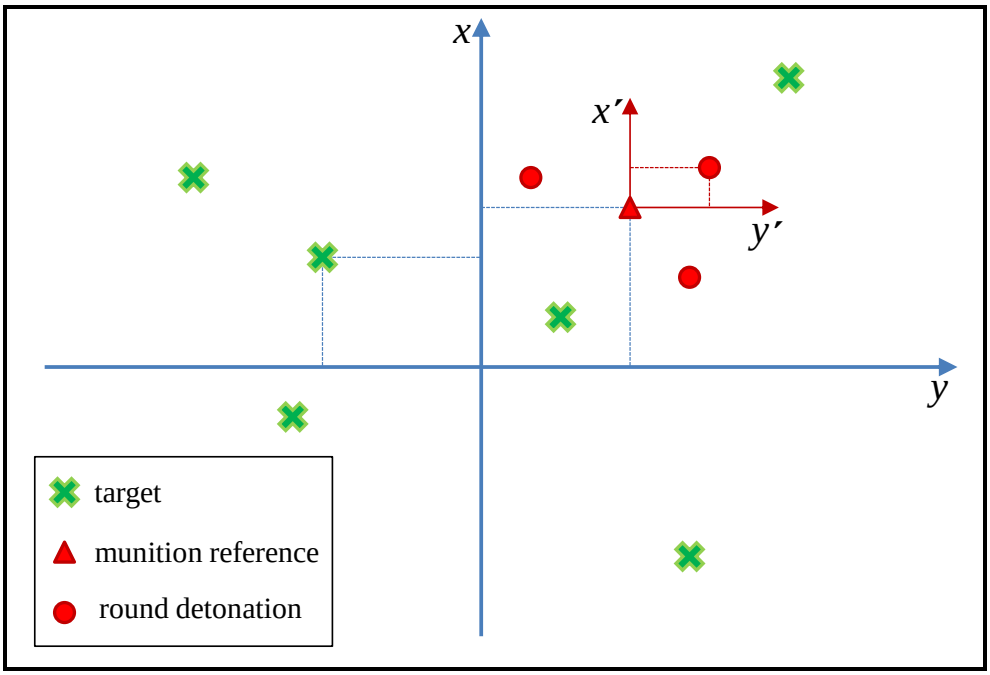

Figure 1. Coordinates used by the yScalable program.

5.1 Target Locations

Target locations are specified in target-location files (see section 6.4) and are referenced to the primary coordinate system. Exactly one personnel target is assumed to exist at each target location. Target-location error is assumed to be zero.

5.2 Munition-Reference Locations

Munition-reference locations specify the origins of secondary Cartesian coordinate systems. They allow round-detonation locations to be varied without disturbing overall round-detonation patterns. Munition-reference locations are obtained by combining the munition aimpoints that are found in munition-aimpoint files (see section 6.2) with a randomly chosen offset. Offsets are selected by drawing two normally-distributed random numbers, each with a standard deviation that is uniquely determined by the Circular Error Probable (CEP) for the simulation. CEP values are specified in master-input files (see section 6.1). Equation 1 shows the relationship between standard deviation and CEP.

$$\sigma_x = \sigma_y = \frac{1}{\sqrt{2\ln(2)}} CEP\,. \tag{1}$$

5.3 Round-Detonation Locations

Round-detonation locations specify the detonation points of rounds with respect to munition-reference locations (i.e., they are specified in secondary coordinates). Although there is no explicit error associated with round-detonation locations, their locations are affected by the CEP that is specified for munition aimpoints.

6. Input Files

The yScalable program requires five input files: one master input file and four auxiliary files.

6.1 Master-Input File

The master-input file specifies all scenario parameters, as well as the names of four auxiliary files. Table 1 gives a line-by-line description of the master-input file, while figure 2 presents a sample master-input file. Three types of lines are present in the master-input file. Each type is color coded to aid in identification.

Comment Lines: Comment lines must be present in the master-input file, even if they are simply blank lines. They can contain any information that the user deems helpful. They can begin with any character.

Numeric Parameters: Lines containing numeric parameters must begin with a number. Comments are allowed to follow the numerical values but must be separated from numeric values by one or more spaces. No special symbol is required to denote a comment.

Auxiliary-File Names: Lines containing the names of auxiliary files must begin with the path and name of an auxiliary file. Comments are not allowed.

Table 1. Master-input file.

Line	Description
1	comment line
2	comment line
3	This line specifies the number of calls used to initialize the pseudorandom number generator, thus allowing the user to specify the starting point for the sequence of pseudorandom numbers that is generated by the pseudo-random number generator.
4	This line specifies the Circular Error Probable (CEP) for all munitions (note that CEP for rounds is 0.0).
5	This line specifies the number of iterations that will be performed in the Monte Carlo simulation.
6	This line specifies the number of barrages per iteration.
7	This line specifies the number of munitions per barrage.
8	comment line
9	This line specifies the path and name of a munition-location file.
10	This line specifies the number of rounds per munition.
11	comment line
12	This line specifies the path and name of a round-detonation file.
13	This line specifies the number of targets in the target set.
14	comment line
15	This line specifies the path and name of a target-location file.
16	comment line
17	This line specifies the path and name of a PI-grid file.

```
# This file was created automatically on 2012-01-23 at 12:47:34
# using the BillModIn.exe program.
40000     # Number of calls to initialize random number generator
10.00     # CEP for munitiions (CEP for rounds is 0)
100000    # Number of iterations
1         # Number of barrages per iteration
3         # Number of munitions per barrage
# Next line is the munitions location file
munition_files\11.mun
1         # Number of rounds per munition
# Next line is the location of the rounds within a single munition
weapon_directories\155\155.rnd
30        # Number of targets
# Next line is the IC formation locations file
target_files\tgt1platoonv30man.txt
# Next line is the Pi grid table for the given range
weapon_directories\155\polarbaseH6.a5m
```

Figure 2. Master-input file.

6.2 Munition-Aimpoint File

Munition-aimpoint files specify the munition aimpoints that are described in section 5.2. Recall that munition aimpoints are unperturbed munition-reference locations. Figure 3 presents a sample munition-aimpoint file. Each line of the file represents the aimpoint of a munition in the primary coordinate system. Two entries are required per line. The first entry is the x-coordinate of the munition, the second entry is the y-coordinate. Note that the aimpoint file may contain more aimpoints than the number of munitions specified in the master-input file. If that is the case, then munition aimpoints are taken from the beginning of the munition aimpoint file until the required number of aimpoints has been reached.

```
0.0 0.0
10.0 10.0
-10.0 -10.0
```

Figure 3. Sample munition-location file.

6.3 Round-Detonation File

Round-detonation files specify the detonation locations of rounds *with respect to munition-reference locations*.

Figure 4 presents a sample round-detonation file. Each line of the file represents the detonation location of a round in a secondary coordinate system. Two entries are required per line. The first entry is the x-coordinate, the second entry is the y-coordinate. Note that the round-detonation file may contain more detonation points than the number of rounds specified in the master-input file. If that is the case, then round-detonation points are taken from the beginning of the round-detonation file until the required number of detonation points has been reached.

```
0 0
1 1
1 -1
-1 1
-1 -1
```

Figure 4. Sample round-detonation file.

5.4 Target-Location File

Target-location files specify the locations of all personnel targets. Each line of the file represents the location of a single target in the primary coordinate system. Two entries are required per line. The first entry is the x-coordinate, the second entry is the y-coordinate. Note that the target-location file may contain more target locations than the number of targets specified in the master-input file. If that is the case, then target locations are taken from the beginning of the target-location file until the required number of target locations has been reached.

```
0 0
-12.7 0
12.7 0
-7.16 6.36
7.16 6.36
```

Figure 5. Sample target-location file.

6.5 PI-Grid File

PI-grid files contain tabulated PI values as a function of distance from a detonation and an azimuthal angle that is referenced from a fragmentation round's velocity vector at the time of impact. PI-grid files can be produced by several different sources including CASRED,[2] MUVES-S2,[3] and JMAE.[4] Figure 6 presents a sample PI-grid file.

[2]Butler, S. *Casualty Reduction (CASRED) Model - Source Code Documentation*; U.S. Army Materiel Systems Analysis Activity, 1975.

[3]Yager, R. J. *A C++ Postprocessor for Modifying Probability-of-Kill Grids Created by the Joint Mean Area of Effects (JMAE) Model*; ARL-TR-4982; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2009.

[4]Yager, R. J. *A C++ Postprocessor for Converting MUVES-S2 Vehicle-Centered Lethality Grids to Detonation-Centered Grids*; ARL-TR-5448; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, 2011.

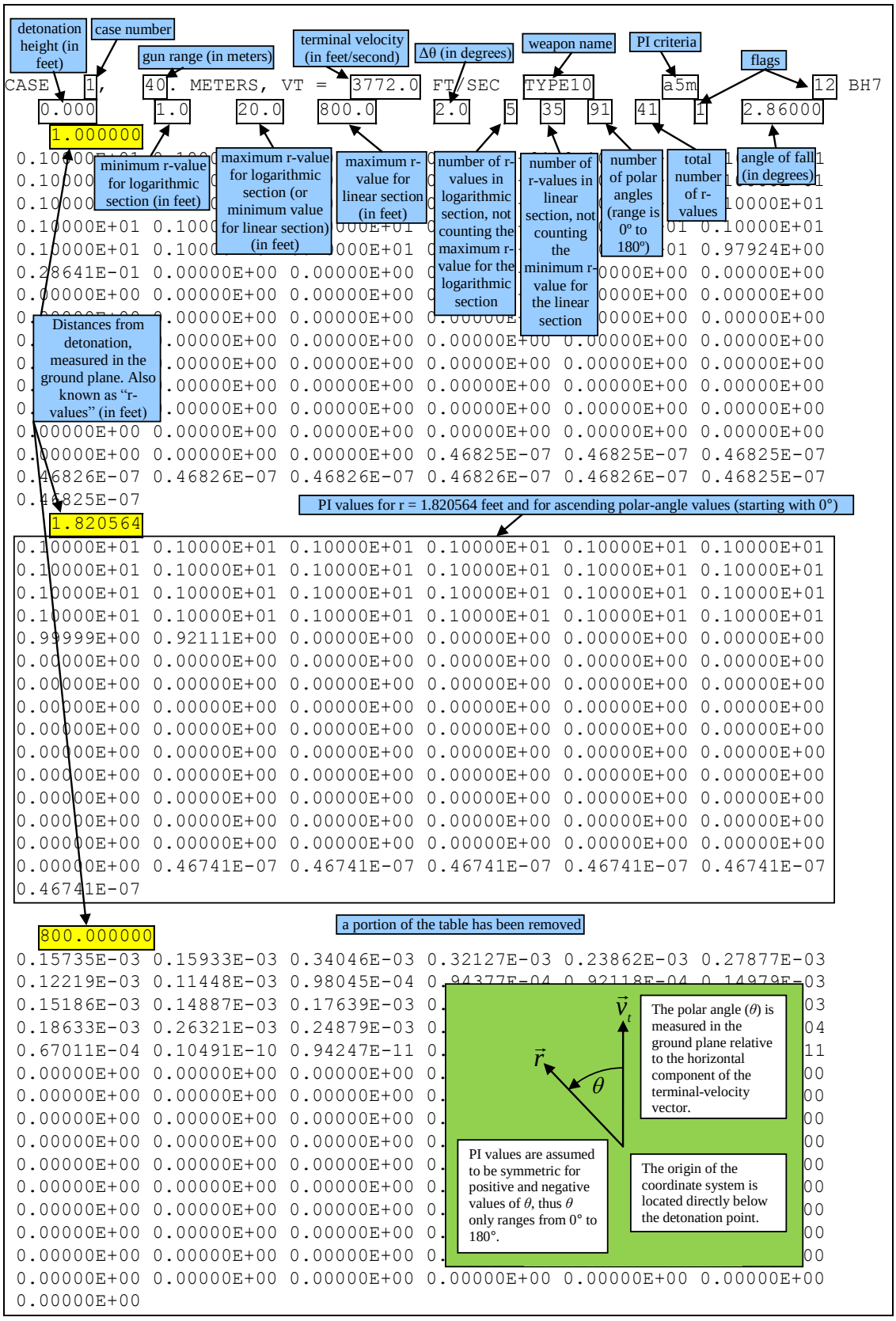

Figure 6. Sample annotated PI-grid file.

7. Output Files

Each time that the yScalable program is run, a single output file is created. Output files contain the date and time that the file was created, a restatement of the input parameters and auxiliary filenames contained in the master-input file, and four values that represent a summary of the scenario. Figure 7 presents a sample yScalable output file.

```
Date Run: 01/24/12,Time: 08:33:15
Input File: input_files_temp\SampleMI.txt
Target Formation Location File: target_files\SampleTarget.txt
Munition Location File: munition_files\Sample.mun
Round Location File: weapon_directories\Sample.rnd
Using P(i) file: weapon_directories\Sample.a5m

The average number of incapacitations per iteration is: 30.0000
The average Pi is: 0.9500
The probability of incapacitating 100% of targets is: 0.9500
The average maximum distance from origin to impact is: 30.0000

The number of iterations is: 100000
The number of barrages per iteration is: 5
The number of munitions per barrage is: 3
The number of rounds per munition is: 1
```

Figure 7. Sample output file (with simulated values).

7.1 Average Number of Incapacitations Per Iteration

In order to calculate the average number of incapacitations per iteration, a list that tracks whether or not each target has been incapacited in a given Monte Carlo iteration is used. At the end of each iteration, the total number of incapacitations is counted. This method ensures that, for a given iteration, no target is counted as having been incapacitated more than once.

7.2 Average PI

The average PI is calculated by dividing the average number of incapacitations per iteration by the number of targets.

7.3 Probability of Incapacitating 100% of Targets

For each iteration, it is determined whether or not all of the targets were incapacitated. The probability of incapacitating 100% of targets is then calculated by dividing the number of times that all of the targets were incapacitated by the number of iterations.

7.4 Average Maximum Distance From Origin to Impact

For each iteration, the distance from the origin to the impact point of each round is calculated, then the maximum distance is stored. These values are used to calculate the average maximum distance from the origin to impact.

---

8. Running yScalable

The yScalable program can be run directly from the command line or from a script. To start the program type the name of the compiled executable file followed by the name of the master-input file followed by the name of the output file (e.g., "y_scalable.exe sample_input.txt sample_output.txt").

INTENTIONALLY LEFT BLANK.

Appendix. Code

This appendix appears in its original form, without editorial change.

y_scalable_main.cc

```
/*****************************************************************************
*                                yScalable                                   *
*               Rob Yager, 10JAN2011 (last updated 24JAN2012)                *
*****************************************************************************/
#include "oberle_aimpoint_tools.h"
#include <vector>
/*****************************STRUCTS AND TYPEDEFS*****************************/
struct PAIR{double x , y;};//.................................x-y ordered pair
typedef vector<PAIR> PAIRS;//................................1D array of PAIRs
/***********************GLOBAL, USER-DEFINED PARAMETERS***********************/
char cMunsInBar[150];//..................file containing aim points of munitions
char cRdsInMun[150];//.....................file containing aim points of rounds
char cTargetLocation[150];//......file containing locations of personnel targets
char cPiTable1[150];//................................file containing Pi table
double dPiTable1[200][92]={0};//.....................................Pi table
int iRandu;//..........number of calls to initialize the random-number generator
int iterations;//.........................total number of scenarios to be ran
int iNumBar;//............................number of barrages in each iteration
int iNumMun;//............................number of munitions in each barrage
int iNumRds;//...........................number of rounds within each munition
int iNumTargets;//................................number of personnel targets
int iLinesPiTable1;//.........................number of lines in the Pi table
double dcep;//.................................................CEP for munitions
PAIRS MUNS;//............locations of aim points for munitions (from cMunsInBar)
PAIRS RNDS;//.........offset locations of aim points for rounds (from cRdsInMun)
PAIRS TARS;//..............locations of personnel targets (from cTargetLocation)
/*****************************GLOBAL VARIABLES*****************************/
double avg_incaps=0;//.......................average number of incapacitations
double avg_r_max=0;//............average maximum distance from origin to impact
double prob_100=0;//..............probability of incapacitating 100% of targets
/*****************************FUNCTION PROTOTYPES*****************************/
void get_vars(char* argv[]);//..................reads variables from input files
double rndm(double avg,double std);//.........overload of Bill's rndm() function
bool incap(PAIR target,PAIR detonation);//.......output of true => incapacitated
void output(char* argv[]);//..............................writes an output file
/************************************MAIN************************************/
int main(int argc,char* argv[]){
  //-------------------------READ THE INPUT FILE------------------------------
  get_vars(argv);
  //---------------INITIIALIZE THE RANDOM NUMBER GENERATOR---------------------
  for(int i=0;i<iRandu;++i) rndm(0,1);
  //--------CALCULATE THE AVERAGE NUMBER OF INCAPACITATIONS PER BARRAGE---------
  double sigma=dcep*0.8493218002880191;//.....1/sqrt(2*ln(2))=0.8493218002880191
  for(int i=0;i<iterations;++i){//..................loop through all iterations
    double r_max=0;//.....maximum distance (per iteration) from origin to impact
    vector <bool> INCAPS(iNumTargets , false);//..............incapacitated list
    for(int i2=0;i2<iNumBar;++i2){//......loop through all barrages in iteration
      for(int j=0;j<iNumMun;++j){//........loop through all munitions in barrage
        //- - - - - - calculate the location of the jth munition - - - - - - - -
        PAIR MUN = {MUNS[j].x + rndm(0,sigma) , MUNS[j].y + rndm(0,sigma)};
        for(int k=0;k<iNumRds;++k){//........loop through all rounds in munition
          //- - - - - - calculate the location of the kth round- - - - - - - - -
          PAIR RND = {MUN.x + RNDS[k].x , MUN.y + RNDS[k].y};//round CEP is zero
          r_max=max(r_max,hypot(RND.x,RND.y));
```

```
        for(int l=0;l<iNumTargets;++l)//..............loop through all targets
          if(!INCAPS[l]) INCAPS[l] = incap(TARS[l],RND);}}}//...incapacitated?
    //- - - - - - - - maintain cumulative moving averages- - - - - - - - - - - -
    int n=0;//........n is the number of incapacitations for the current barrage
    for(int l=0;l<iNumTargets;++l)if(INCAPS[l])n++;//...................tally n
    avg_incaps=(avg_incaps*i+n)/(i+1);//....cumulative avg. # of incapacitations
    avg_r_max=(avg_r_max*i+r_max)/(i+1);//average max dist from origin to impact
    prob_100=((n==iNumTargets?1:0)+i*prob_100)/(i+1);//cumul. avg of 100% incaps
    if(i%1000==999)printf("\rIn iteration number: %d",i+1);}
  //-----------------------WRITE THE OUTPUT FILE-------------------------------
  output(argv);
  return 0;
}/*******ROB*YAGER**************10JAN2012********(LAST*UPDATED*11JAN2012)******/
void get_vars(char* argv[]){
  int i;
  ifstream inFile1;
  char cLine[150];
  inFile1.open(argv[1], ios::in  | ios::binary);//Input File Name
  if(!inFile1)
  {
      cerr << "Cannot open input file: " << argv[1] << endl;
      cerr << "Program is terminating." << endl;
      exit(1);
  }
  inFile1.getline(cLine,149);    //***** Reads comment line #1
  inFile1.getline(cLine,149);    //***** Reads comment line #2
  inFile1 >> iRandu ;            //***** Random Number initializer
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1>>dcep;                 //***** CEP for munitions
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1>>iterations;           //***** number of iterations
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1>>iNumBar;              //***** number of barrages
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1>>iNumMun;              //***** number of munitions
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1.getline(cLine,149);    //***** Reads comment line #3
  inFile1.getline(cMunsInBar,149);//***** Read location of munition in barrage
    //***** Need to remove any carriage returns from string and replace with \0
    for(i = 0; cMunsInBar[i]; i++)
    {
        if((int)cMunsInBar[i] == 13) cMunsInBar[i] = '\0';
    }
  inFile1>>iNumRds;              //***** number of Rounds
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1.getline(cLine,149);    //***** Reads comment line #4
  inFile1.getline(cRdsInMun,149);  //***** Read location of rounds in munition
    //***** Need to remove any carriage returns from string and replace with \0
    for(i = 0; cRdsInMun[i]; i++)
    {
        if((int)cRdsInMun[i] == 13) cRdsInMun[i] = '\0';
    }
  inFile1>>iNumTargets;          //***** number of Targets
  inFile1.getline(cLine,149);     //***** Read rest of line
  inFile1.getline(cLine,149);     //***** Read comment line #5
  inFile1.getline(cTargetLocation,149);  // Read Target Location Table file name
    //***** Need to remove any carriage returns from string and replace with \0
    for(i = 0; cTargetLocation[i]; i++)
```

```
  {
      if((int)cTargetLocation[i] == 13) cTargetLocation[i] = '\0';
  }
inFile1.getline(cLine,149);    //***** Read comment line #6
inFile1.getline(cPiTable1,149);  //***** Read Pi Table file name
  //***** Need to remove any carriage returns from string and replace with \0
  for(i = 0; cPiTable1[i]; i++)
  {
      if((int)cPiTable1[i] == 13) cPiTable1[i] = '\0';
  }
inFile1.close();
//*************************************************************
//***** Read munition location within a barrage ****************
//*************************************************************
inFile1.open(cMunsInBar,ios::in | ios::binary);
if(!inFile1)
  {
      cerr << "Cannot open input file: " << cMunsInBar << endl;
      cerr << "Program is terminating." << endl;
      exit(1);
  }
for(i = 0; i < iNumMun; i++)
{
  MUNS.push_back(PAIR());
  inFile1 >> MUNS[i].x >> MUNS[i].y;
}
inFile1.close();
//*************************************************************
//***** Read round location within a mnition ******************
//*************************************************************
inFile1.open(cRdsInMun,ios::in | ios::binary);
if(!inFile1)
  {
      cerr << "Cannot open input file: " << cRdsInMun << endl;
      cerr << "Program is terminating." << endl;
      exit(1);
  }
for(i = 0; i < iNumRds; i++)
{
  RNDS.push_back(PAIR());
  inFile1 >> RNDS[i].x >> RNDS[i].y;
}
inFile1.close();
//*************************************************************
//******** Read Target Locations ******************************
//*************************************************************
inFile1.open(cTargetLocation,ios::in | ios::binary);
if(!inFile1)
  {
      cerr << "Cannot open input file: " << cTargetLocation << endl;
      cerr << "Program is terminating." << endl;
      exit(1);
  }
for(i = 0; i < iNumTargets; i++)
{
  TARS.push_back(PAIR());
  inFile1 >> TARS[i].x >> TARS[i].y;
}
```

```
  inFile1.close();
  //*************************************************************
  //***** The Pi table is read ************************************
  //*************************************************************
  iLinesPiTable1 = ReadPiTables(cPiTable1,dPiTable1);
  //*************************************************************
}/*******ROB*YAGER*************10JAN2012********(LAST*UPDATED*11JAN2012)******/
double rndm(double avg,double std){//.........overload of Bill's rndm() function
  long double x;
  rndm(avg,std,x);
  return x;
}/*******ROB*YAGER*************10JAN2012********(LAST*UPDATED*10JAN2012)******/
bool incap(PAIR target,PAIR detonation){
  double distance,angle;
  DetermineDistanceAngle(target.x,target.y,detonation.x,detonation.y,
                         distance,angle);
  return bDeterminePiV(distance,angle,iLinesPiTable1,dPiTable1);
}/*******ROB*YAGER*************10JAN2012********(LAST*UPDATED*10JAN2012)******/
void output(char* argv[]){
  ofstream outFile;
  outFile.open(argv[2], ios::out | ios::binary);//Output File Name
  char dateStr [9];
  char timeStr [9];
  _strdate_s( dateStr);
  _strtime_s( timeStr );
  outFile << "Date Run: " << dateStr << ",Time: " << timeStr << endl;
  outFile << "Input File: " << argv[1] << endl;
  outFile << "Target Formation Location File: " << cTargetLocation << endl;
  outFile << "Munition Location File: " << cMunsInBar << endl;
  outFile << "Round Location File: " << cRdsInMun << endl;
  outFile << "Using P(i) file: " << cPiTable1 << endl << endl;
  outFile.width(6);
  outFile.precision(4);
  outFile.setf(cout.fixed);
  outFile << "The average number of incapacitations per iteration is: "
    << avg_incaps << endl;
  outFile << "The average Pi is: " << avg_incaps/iNumTargets << endl;
  outFile << "The probability of incapacitating 100% of targets is: "
    << prob_100 << endl;
  outFile << "The average maximum distance from origin to impact is: "
    << avg_r_max << endl << endl;
  outFile << "The number of iterations is: " << iterations << endl;
  outFile << "The number of barrages per iteration is: " << iNumBar << endl;
  outFile << "The number of munitions per barrage is: " << iNumMun << endl;
  outFile << "The number of rounds per munition is: " << iNumRds << endl;
  outFile.close();
}/*******ROB*YAGER*************11JAN2012********(LAST*UPDATED*11JAN2012)******/
```

oberle_aimpoint_tools.h

```
#ifndef OBERLE_AIMPOINT_TOOLS_H_//..............#define guard (yager, 2012-01-10)
#define OBERLE_AIMPOINT_TOOLS_H_//..............#define guard (yager, 2012-01-10)
#include "Oberle_Nov_2010.h"

//***** Global Varialbes and Fcn Prototypes Used in the Progam

bool bCheckPi;
```

```
int ReadPiTables(char *,double [200][92]); // Returns number of lines in table
bool bDeterminePiV(double,double,int,double [200][92]);
void DetermineDistanceAngle(double,double,double,double, double &, double &);

int ReadPiTables(char * fileName1, double dPiTab[200][92])
{
    char cLine[120];
    double dHeight, dMaxDist, dDegree, dAngle, x, y;
    int i, j, k, iNum, iNumB,iFileCount = 0;


    ifstream F2P(fileName1,ios::in | ios::binary);
  if(!F2P)
        {
               cerr << "Cannot open input file: " << fileName1 << endl;
               cerr << "Program is terminating." << endl;
               exit(1);
        }
    F2P.getline(cLine,120); //Reads the header line - one time only
    if(bCheckPi) cout << cLine << endl;
    F2P >> dHeight >> x >> y >> dMaxDist >> dDegree >> i >> j >> iNum >> iNumB >> k
>> dAngle;
    F2P.getline(cLine,120);
  if(bCheckPi)
  {
        cout << "iNum: " << iNum << endl;
        cout << "iNumB: " << iNumB << endl;
  }

    for(i = 0; i < iNumB; i++) // Number of distance blocks
    {
        F2P >> x; // Read the distance from impact point
        dPiTab[i][0] = x; // Write distance from impact point to table
        for(j = 0; j < iNum; j++)
        {
            F2P >> y;
            dPiTab[i][j+1] = y; // Write Pi for angle
        }
     }
    F2P.close();

  if(bCheckPi)
  {
        for(j=0;j<iNumB;j++)
        cout << "Line :" << j << "\t" << dPiTab[j][0] << "\t" << dPiTab[j][1] <<
"\t" << dPiTab[j][2] << "\t" << dPiTab[j][iNum-1] << "\t" << dPiTab[j][iNum] << endl;
        system("pause");
  }

    return iNumB;
}
bool bDeterminePiV(double r, double theta,int iLines,double dPiTab[200][92])
{
    int iLowAngle,iLowRadius,i;
    double dAngle,dProbI;
    double A[4][3];
    long double dR;
```

```
    bool iFlag = false;

    //***** Start by determining the index in *********************
    //***** in the table for the angle ***************************
    //***** Convert angle to degrees *****************************

    dAngle = dRad2Deg(theta); //***** convert angle from radians to degrees
    iLowAngle = (int) floor(dAngle/2 + 1); //***** Angle index is iLowAngle and
iLowAngle+1
    if(iLowAngle == 91) iLowAngle = 90; //***** Just in case angle is 180 degrees


    //***** Determine that range is in table
    if( (r < 0) || (r > dPiTab[iLines-1][0])) //Range not in table.
    {
        //cout << "Returned a false on distance\n";
        return false;
         //return 0.0;

    }

  iFlag = false;

    if ( r < dPiTab[0][0])  // Positive but less than min distance
  {
        r = dPiTab[0][0];
  }

    if(!iFlag)
        {
        for(i = 0; i < iLines; i++)
        {
            if(r > dPiTab[i][0])
            {
                continue;
            }
            else
            {
                iLowRadius = i-1; //***** Radius index is iLowRadius and iLowRadius+1
                break;
            }
        }
        if(iLowRadius == -1) iLowRadius = 0;
    }

    if(!iFlag)
    {
        A[0][0] = (iLowAngle -1) * 2; //Angle
        A[0][1] = dPiTab[iLowRadius][0]; //Radius
        A[0][2] = dPiTab[iLowRadius][iLowAngle]; //Pi

        A[1][0] = iLowAngle * 2;
        A[1][1] = dPiTab[iLowRadius][0];
        A[1][2] = dPiTab[iLowRadius][iLowAngle+1];

        A[2][0] = iLowAngle * 2;
        A[2][1] = dPiTab[iLowRadius+1][0];
        A[2][2] = dPiTab[iLowRadius+1][iLowAngle+1];
```

```
        A[3][0] = (iLowAngle -1) * 2;
        A[3][1] = dPiTab[iLowRadius+1][0];
        A[3][2] = dPiTab[iLowRadius+1][iLowAngle];
    }
    else
    {
        A[0][0] = (iLowAngle -1) * 2; //Angle
        A[0][1] = 0; //Radius
        A[0][2] = 1; //dPiTab[iLowRadius][iLowAngle]; //Pi

        A[1][0] = iLowAngle * 2;
        A[1][1] = 0;
        A[1][2] = 1; //dPiTab[iLowRadius][iLowAngle+1];

        A[2][0] = iLowAngle * 2;
        A[2][1] = dPiTab[0][0];
        A[2][2] = dPiTab[0][iLowAngle+1];

        A[3][0] = (iLowAngle -1) * 2;
        A[3][1] = dPiTab[0][0];
        A[3][2] = dPiTab[0][iLowAngle];
    }

    dProbI = dBilinear(dAngle,r,A);

    randu(0.0,1.0,dR);
    //cout << "the random number is: " << " " << dR << endl;
    if(dR <= dProbI) return true;

    return false;

  //return dProbI;
}



void DetermineDistanceAngle(double dXTemp, double dYTemp,double dX,double dY,double
&distance,double &angle)
{
  double dTemp1,dTemp2,dTemp3,dTemp4,distanceM;
  double TrueAngle;

  dTemp1 = dXTemp - dX;
  dTemp2 = dYTemp - dY;
  //the vector (dTemp1,dTemp2) is the vector from the impact point to the target i

  dTemp3 = (dXTemp - dX)*(dXTemp - dX);//square of the difference in x poisition (x
is range)
  dTemp4 = (dYTemp - dY)*(dYTemp - dY);//square of the difference in y poisition (y
is azimuth)
  dTemp4 = sqrt(dTemp3+dTemp4);//distance between the impact point and target i in
meters
  distanceM = dTemp4;//distance in meters

  //***** now the vector <1,0> is the direction of the round; i.e., have no yaw
information so
  //***** assuming round is flying in a straight line parallel to the range axis.
```

```
   //***** Thus, the dot product between the vector <1,0> and the vector
   //***** from the impact point to the target location is dXTemp1.
   dTemp4 = dTemp1/distanceM;//dot product divided by distance no units

   angle = acos(dTemp4); //Angle between vectors; this value is between 0 and Pi
[0,180]
   //***** degrees; this value is in radians; 0 degrees is along the range axis;

   if(dYTemp < dY)// gives the true angle between the impact pt and target;
          //***** will use negative angles between 0 and 180 instead of 180 to 359
          //***** informational only
   {
          TrueAngle = - angle;
   }
   distance = distanceM/.3048;// Convert from meters to feet (could mul by
3.280839895)
   if(distance <= 0) distance =.0001;

   return;

}

#endif//.................................end #define guard (yager, 2012-01-10)
```

NO. OF COPIES	ORGANIZATION
1 (PDF only)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218
1	DIRECTOR US ARMY RESEARCH LAB IMAL HRA 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	US ARMY ARDEC AMSRD AAR AIS SA D ERICSON BLDG 12 PICATINNY ARSENAL NJ 07806-5000
1	US ARMY ARDEC RDAR MEE W A DANIELS BLDG 3022 PICATINNY ARSENAL NJ 07806

ABERDEEN PROVING GROUND

NO. OF COPIES	ORGANIZATION
22 (20 HC 2 CD)	DIR USARL RDRL WM P BAKER P PLOSTINS RDRL WML M ZOLTOSKI RDRL WML A P BUTLER W OBERLE (1 CD) C PATTERSON R PEARSON L STROHM R YAGER (4 HC, 1 CD) RDRL WML B J MORRIS RDRL WML C K MCNESBY RDRL WML D R BEYER RDRL WML E P WEINACHT RDRL WML F T BROWN D LYON RDRL WML H M FERMEN-COKER J NEWILL B SORENSEN